initial code from hw2

In [22]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from tqdm import tqdm
from ucimlrepo import fetch_ucirepo
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler, OneHotEncoder
```

In [28]:
```
# fetch dataset
wine = fetch_ucirepo(id=109)

# data (as pandas dataframes)
X_original = wine.data.features
y_original = wine.data.targets

y = OneHotEncoder().fit_transform(y_original).toarray()

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_original, y, test_size

# normalize each column in the data
scaler = RobustScaler()
X_train = scaler.fit(X_train).transform(X_train)
X_test = scaler.transform(X_test)
```

In [29]:
```
# Sigmoid activation function and its derivative
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)
```

In [30]:
```
def forward(X, W1, b1, W2, b2):
    hidden_layer_input = X @ W1 + b1
    hidden_layer_output = sigmoid(hidden_layer_input)

    output_layer_input = hidden_layer_output @ W2 + b2
    predicted_output = sigmoid(output_layer_input)

    return hidden_layer_output, predicted_output

# Use the forward function in the training loop
def train_mlp(X, y, num_of_hidden_layers=8, learning_rate=0.1, epochs=100000
    # Initialize parameters
    input_layer_neurons = X.shape[1]            # Number of input neurons (x
```

```
    hidden_layer_neurons = num_of_hidden_layers # Number of hidden neurons
    output_neurons = y.shape[1]                 # Output neuron (binary clas

    # Randomly initialize weights and biases
    np.random.seed(seed)
    W1 = np.random.uniform(size=(input_layer_neurons, hidden_layer_neurons))
    b1 = np.random.uniform(size=(1, hidden_layer_neurons))
    W2 = np.random.uniform(size=(hidden_layer_neurons, output_neurons))
    b2 = np.random.uniform(size=(1, output_neurons))

    for epoch in tqdm(range(epochs), desc="Training MLP"):
        # Forward Propagation
        hidden_layer_output, predicted_output = forward(X, W1, b1, W2, b2)

        # Backpropagation
        error = y - predicted_output
        d_predicted_output = error * sigmoid_derivative(predicted_output)

        error_hidden_layer = d_predicted_output @ (W2.T)
        d_hidden_layer = error_hidden_layer * sigmoid_derivative(hidden_laye

        # Updating Weights and Biases
        W2 += hidden_layer_output.T @ (d_predicted_output) * learning_rate
        b2 += np.sum(d_predicted_output, axis=0, keepdims=True) * learning_r
        W1 += X.T @ (d_hidden_layer) * learning_rate
        b1 += np.sum(d_hidden_layer, axis=0, keepdims=True) * learning_rate

        # Optionally print loss at intervals
        if epoch % 1000 == 0 and verbose:
            loss = np.mean(np.square(y - predicted_output))
            print(f'Epoch {epoch}, Loss: {loss}')

    hidden_layer_output, predicted_output = forward(X, W1, b1, W2, b2)
    loss = np.mean(np.square(y - predicted_output))
    print(f'Final Epoch {epoch}, Loss: {loss}')

    return W1, b1, W2, b2, predicted_output

# Call the function to train the MLP
W1, b1, W2, b2, predicted_output = train_mlp(X_train, y_train, verbose=False
```

```
Training MLP: 100%|██████████| 100000/100000 [00:03<00:00, 29343.70it/s]
Final Epoch 99999, Loss: 8.452705833625564e-07
```

In [31]:
```
from sklearn.metrics import confusion_matrix, classification_report, accurac

# Define a function to test multiple hyperparameters
def test_hyperparams(X, y, hidden_layers_list, learning_rates, epochs_list):
    best_accuracy = 0
    best_params = {}
```

```
    for hidden_layers in hidden_layers_list:
        for lr in learning_rates:
            for epochs in epochs_list:
                print(f"Testing with {hidden_layers} hidden layers and learr
                W1, b1, W2, b2, predicted_output = train_mlp(X, y, num_of_hi
                hidden_layer_output, predicted_output = forward(X_test, W1,
                y_pred_test = np.argmax(predicted_output, axis=1)
                y_true_test = np.argmax(y_test, axis=1)
                accuracy = accuracy_score(y_true_test, y_pred_test)
                print(f"Accuracy: {accuracy * 100:.2f}%")
                conf_matrix = confusion_matrix(y_true_test, y_pred_test)
                print("Confusion Matrix:")
                print(conf_matrix)
                if accuracy > best_accuracy:
                    best_accuracy = accuracy
                    best_params = {'hidden_layers': hidden_layers, 'learning
    return best_params, best_accuracy

# Define hyperparameters to test
hidden_layers_list = [2, 4, 8]
learning_rates = [0.01, 0.1, 0.5]
epochs_list = [100, 1000, 10000]

# Test hyperparameters
best_params, best_accuracy = test_hyperparams(X, y, hidden_layers_list, lear
print(f"Best Accuracy: {best_accuracy * 100:.2f}%")
```

```
Testing with 2 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 15264.78it/s]
Final Epoch 99, Loss: 0.08637913695802092
Accuracy: 98.31%
Confusion Matrix:
[[20  0  0]
 [ 0 23  1]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 26886.74it/s]
Final Epoch 999, Loss: 0.0032583243909230885
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 35861.96it/s]
```

```
Final Epoch 9999, Loss: 0.00021905212155582036
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 29259.18it/s]
Final Epoch 99, Loss: 0.0032511765985438943
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 17497.77it/s]
Final Epoch 999, Loss: 0.00021851236288562893
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 33416.99it/s]
Final Epoch 9999, Loss: 1.9270135908262373e-05
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 2 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 27769.49it/s]
Final Epoch 99, Loss: 0.20053076322908775
Accuracy: 40.68%
Confusion Matrix:
[[ 0 20  0]
 [ 0 24  0]
 [ 0 15  0]]
Testing with 2 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 34177.56it/s]
Final Epoch 999, Loss: 0.08990598442207061
Accuracy: 74.58%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [14  1  0]]
Testing with 2 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 35289.79it/s]
```

```
Final Epoch 9999, Loss: 4.054976332658819e-06
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 23736.86it/s]
Final Epoch 99, Loss: 0.05677250832227215
Accuracy: 96.61%
Confusion Matrix:
[[19  1  0]
 [ 1 23  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 27431.14it/s]
Final Epoch 999, Loss: 0.0019930615004680337
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 28166.72it/s]
Final Epoch 9999, Loss: 0.00012332551816770718
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 23132.05it/s]
Final Epoch 99, Loss: 0.0019847662397258214
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 27133.55it/s]
Final Epoch 999, Loss: 0.0001231839851077531
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 28556.67it/s]
```

```
Final Epoch 9999, Loss: 1.0056136006763757e-05
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 23546.31it/s]
Final Epoch 99, Loss: 0.0003648941074179784
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 28021.43it/s]
Final Epoch 999, Loss: 2.9060297822402417e-05
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 4 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 27106.14it/s]
Final Epoch 9999, Loss: 2.9954284200671434e-06
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 21082.20it/s]
Final Epoch 99, Loss: 0.04814967866637947
Accuracy: 98.31%
Confusion Matrix:
[[19  1  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 24023.60it/s]
Final Epoch 999, Loss: 0.001913400264254685
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.01
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 24834.96it/s]
```

```
Final Epoch 9999, Loss: 0.00010454723733226173
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 21472.91it/s]
Final Epoch 99, Loss: 0.0022276413058105616
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 24221.29it/s]
Final Epoch 999, Loss: 0.00010634680415606409
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.1
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 24845.34it/s]
Final Epoch 9999, Loss: 7.837299353067711e-06
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 100/100 [00:00<00:00, 21725.39it/s]
Final Epoch 99, Loss: 0.0003111427723735155
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 1000/1000 [00:00<00:00, 24897.18it/s]
Final Epoch 999, Loss: 2.7530480693615916e-05
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Testing with 8 hidden layers and learning rate 0.5
Training MLP: 100%|██████████| 10000/10000 [00:00<00:00, 25231.31it/s]
```

```
Final Epoch 9999, Loss: 2.626514092571934e-06
Accuracy: 100.00%
Confusion Matrix:
[[20  0  0]
 [ 0 24  0]
 [ 0  0 15]]
Best Accuracy: 100.00%
```